

CIÊNCIA DA COMPUTAÇÃO

3ª PERIODO

**“DESENVOLVIMENTO DE SISTEMA PARA ANÁLISE DE
PERFORMANCE DE ALGORITMOS DE ORDENAÇÃO DE
DADOS”**

Índice

Objetivo Trabalho	3
Introdução	4
Referencial Teórico	6
Desenvolvimento	15
Resultados e Discussão	25
Considerações Finais	28
Referências Bibliográficas	29
Código Fonte	30
Fichas APS	43

Objetivo do Trabalho

Esse trabalho tem como objetivo apresentar o que são métodos de ordenação, quais são os mais conhecidos, demonstrar como funcionam, aplicá-los em um programa, obter os resultados de desempenho e discutir sobre eles.

Introdução

Ordenação: Tornar mais simples, rápida e viável a recuperação de uma determinada informação, num conjunto grande de informações.

Algoritmo de ordenação em ciência da computação é um algoritmo que coloca os elementos de uma dada sequência em certa ordem, em outras palavras, efetua sua ordenação completa ou parcial. Existem várias razões para se ordenar uma sequência. Uma delas é a possibilidade de acessar seus dados de modo mais eficiente.

Não existe um método de ordenação considerado universalmente superior a todos os outros. É necessário analisar o problema e, com base nas características dos dados, decidirem qual o método que melhor se aplica a ele.

Alguns aspectos de medida de eficiência: tempo para codificação do programa de ordenação; tempo de máquina para a execução do programa; espaço de memória necessário. Normalmente o tempo gasto é medido pelo número de operações críticas, ao problema, que são efetuadas. Nesse caso as operações críticas são: comparações de chaves; movimentação de registros ou de ponteiros; troca entre dois registros. A medida é tomada pela análise do melhor caso, do pior caso e do caso médio. O resultado é uma fórmula em função de n (número de registros do arquivo).

Na realidade o tempo gasto não depende exclusivamente do tamanho do arquivo, mas também de outros aspectos, por exemplo: se existe uma pré ordenação no arquivo ou não.

Abaixo são apresentados três dos algoritmos de ordenação mais conhecidos.

Bubble Sort

É o método mais intuitivo e simples de ser implementado, porém é o menos eficiente.

Selection Sort

Neste tipo de ordenação, se seleciona o menor elemento do subvetor não ordenado e troca com o primeiro elemento deste subvetor

Quick Sort

O quicksort é um algoritmo de ordenação considerado entre os mais rápidos e eficientes.

O ambiente de desenvolvimento do programa será o NetBeans, utilizando a linguagem Java.

Referencial Teórico

A escolha de um método de ordenação deve – se, a necessidade a qual o software requer, disponibilizando-se de uma biblioteca já estabelecida, porém não fixa, ou seja, podem surgir várias outras, o desenvolvedor pode optar por diversas formas de ordenar. Porém alguns métodos são mais simples e outros mais complexos ou mais eficientes e menos eficientes.

Existem vários desenvolvedores e criadores de métodos e lógicas, mas podemos colocar em destaque James Gosling, que agrega grande conhecimento e influência, assim como experiência, sendo referência na área tecnológica, ajudando a inovar e produzir novos conceitos.

Ordenar corresponde ao processo de rearranjar um conjunto de objeto sem ordem ascendente ou descendente. O objetivo principal da ordenação é facilitar a recuperação posterior de itens do conjunto ordenado. O objetivo principal da ordenação é facilitar a recuperação posterior de itens do conjunto ordenado.

Serão apresentados agora os métodos de ordenação Bubble Sort, Selection Sort e Quick Sort.

Bubble Sort

Bubble sort é um simples e bem conhecido algoritmo de ordenação. Ele é usado mais para prática inicial, para ajudar na introdução aos algoritmos de ordenação. Bubble sort pertence aos $O(n^2)$ algoritmos de ordenação, o que o torna bastante ineficiente para ordenar grandes volumes de dados. Bubble sort é estável e adaptativo.

Algoritmo

1. Compara cada par de elementos adjacentes do início do vetor e, se eles estiverem na ordem reversa, troca eles de lugar.
2. Se ao menos uma troca foi feita, repete o primeiro passo.

Você pode imaginar como se em cada passo grandes bolhas flutuam até a superfície e ficam lá. Em um momento, quando nenhuma bolha flutua, a ordenação para. Vejamos um exemplo do algoritmo ordenando um vetor para termos uma idéia mais clara sobre o bubble sort.

Análise de Complexidade

Em casos de complexidade mediana ou alta, o bubble sort possui desempenho de $O(n^2)$. Assim como, ele efetua $O(n^2)$ trocas nos piores casos. Bubble sort é adaptativo. Isso significa que em vetores quase ordenados seu desempenho estimado é $O(n)$. Evite implementações que não confirmem se o vetor já está ordenado em cada passo (qualquer troca feita). Essa checagem é necessária, em ordem para preservar propriedades adaptativas.

Exemplo:

Ordenando {5, 1, 12, -5, 16} usando bubble sort.

5	1	12	-5	16
---	---	----	----	----

nao ordenado

5	1	12	-5	16
---	---	----	----	----

5 > 1, troca

1	5	12	-5	16
---	---	----	----	----

5 < 12, ok

1	5	12	-5	16
---	---	----	----	----

12 > -5, troca

1	5	-5	12	16
---	---	----	----	----

12 < 16, ok

1	5	-5	12	16
---	---	----	----	----

1 < 5, ok

1	5	-5	12	16
---	---	----	----	----

5 > -5, troca

1	-5	5	12	16
---	----	---	----	----

5 < 12, ok

1	-5	5	12	16
---	----	---	----	----

1 > -5, troca

-5	1	5	12	16
----	---	---	----	----

1 < 5, ok

-5	1	5	12	16
----	---	---	----	----

-5 < 1, ok

-5	1	5	12	16
----	---	---	----	----

ordenado

Ordenação por seleção

Ordenação por seleção é um dos algoritmos de ordenação $O(n^2)$, o que o torna bem ineficiente para ordenar grandes volumes de dados. Ordenação por seleção é notável por sua simples programação e pode superar em performance outros métodos de ordenação em certas situações.

Algoritmo

A ideia do algoritmo é bem simples. O vetor é imaginariamente dividido em duas partes, ordenada e não ordenada. No começo, a ordenada está vazia, enquanto a não ordenada contém todo o vetor. A cada passo, o algoritmo acha o elemento mínimo na parte não ordenada e adiciona ele para o final da parte ordenada. Quando a parte não ordenada se torna vazia, o algoritmo encerra.

Análise de Complexidade

Ordenação por seleção acaba quando a parte não ordenada se torna vazia. Como nós sabemos, em cada passo o número de elementos não ordenados diminui por 1. Portanto, ordenação por seleção faz n passos (n é o número de elementos no vetor) antes de terminar. Cada passo requer achar um mínimo na parte não ordenada. Assim sendo, $n + (n - 1) + (n - 2) + \dots + 1$, resulta em $O(n^2)$ números de comparações. Números de trocas podem variar de 0 (no caso de um vetor ordenado) a $n - 1$ (no caso do vetor ser ordenado em ordem reversa), que resulta em $O(n)$ números de trocas. Em geral a complexidade do algoritmo é $O(n^2)$.

Em fato, ordenação por seleção requer $(n - 1)$ números de trocas na maioria, tornando-o bastante eficiente em situações, onde a operação de obter é significativamente maior do que a opção de ler.

Exemplo:

Ordenando {5, 1, 12, -5, 16, 2, 12, 14} usando ordenação por seleção.

5 1 12 -5 16 2 12 14

5 1 12 -5 16 2 12 14
↑ ↑

-5 1 12 5 16 2 12 14
 ↑
 └─┘

-5 1 12 5 16 2 12 14
 ↑ ↑

-5 1 2 5 16 12 12 14
 ↑
 └─┘

-5 1 2 5 16 12 12 14
 ↑ ↑

-5 1 2 5 12 16 12 14
 ↑ ↑

-5 1 2 5 12 12 16 14
 ↑ ↑

-5 1 2 5 12 12 14 16

Quicksort

Quicksort é um rápido algoritmo de ordenação, que é usado não só para o meio educacional, mas altamente aplicado em prática. Em média, ele tem $O(n \log n)$ de complexidade, tornando o quicksort adequado para ordenar grandes volumes de dados.

A ideia do algoritmo é bem simples e quando você perceber, você poderá escrever quicksort tão rápido quando o bubble sort.

Algoritmo

A estratégia de dividir e conquistar é usada no quicksort. Abaixo o processo de recursão é descrito:

1. Escolha um valor pivô: Nós pegamos o valor do elemento do meio como valor pivô, mas pode ser qualquer valor, desde que esteja entre os valores a serem ordenados.
2. Particione: Reordena os elementos em tal método que, todos os elementos que forem menos que o pivô vão para a parte esquerda do vetor e todos os elementos maiores que o pivô, vão para a parte direita do vetor. Valores iguais ao pivô podem ficar em qualquer parte do vetor. Nota-se que, o vetor pode ser dividido em partes não iguais.
3. Ordene ambas as partes: Aplique o algoritmo quicksort recursivamente para as partes esquerdas e direitas.

Algoritmo de particionamento em detalhe

Há dois índices, i e j , e bem no início do algoritmo i aponta para o primeiro elemento no vetor e j aponta para o último. Em seguida, o algoritmo move o i para frente, até um elemento com valor maior ou igual ao pivô é encontrado. O índice j é movido para trás, até que um elemento com valor inferior ou igual ao pivô seja encontrado. Se $i \leq j$ então eles são trocados e i passa para a próxima posição ($i+1$), j passa para a posição anterior ($j-1$). O algoritmo termina quando i se torna maior do que j .

Após o particionamento, todos os valores anteriores ao elemento i são inferiores ou iguais ao pivô e todos os valores posteriores ao elemento j são maiores ou iguais ao pivô.

Exemplo:

Ordenando {1, 12, 5, 26, 7, 14, 3, 7, 2} usando quicksort.

1 12 5 26 7 14 3 7 2 não ordenado

1 12 5 26 7 14 3 7 2
↑ ↑ ↑
i valor pivô j
valor do pivô = 7

1 12 5 26 7 14 3 7 2
↑ ↑
i j
12 >= 7 >= 2, troca 12 e 2

1 2 5 26 7 14 3 7 12
↑ ↑
i j
26 >= 7 >= 7, troca 26 e 7

1 2 5 7 7 14 3 26 12
↑ ↑
i j
7 >= 7 >= 3, troca 7 e 3

1 2 5 7 3 14 7 26 12
↑ ↑
j i
i > j, para o particionamento

1 2 5 7 3 14 7 26 12
efetua o processo
recursivamente (de
novo)

...

1 2 3 5 7 7 12 14 26
ordenado

Por que funciona?

No algoritmo de particionamento o vetor é dividido em duas partes e todo elemento **a** da parte esquerda é inferior ou igual a todos os elementos **b** da parte direita. **A** e **b** também satisfazem $a \leq \text{pivô} \leq b$. Após o encerramento do processo de recursão, ambas as partes se tornam ordenadas e, levando em conta todos os argumentos apresentados acima, todo o vetor é ordenado.

Análise da Complexidade

Em média, o quicksort possui $O(n \log n)$ de complexidade, mas nos piores casos, o quicksort roda em $O(n^2)$ de tempo, mas nos mais "práticos" dados ele funciona bem e superior à outros $O(n \log n)$ algoritmos de ordenação.

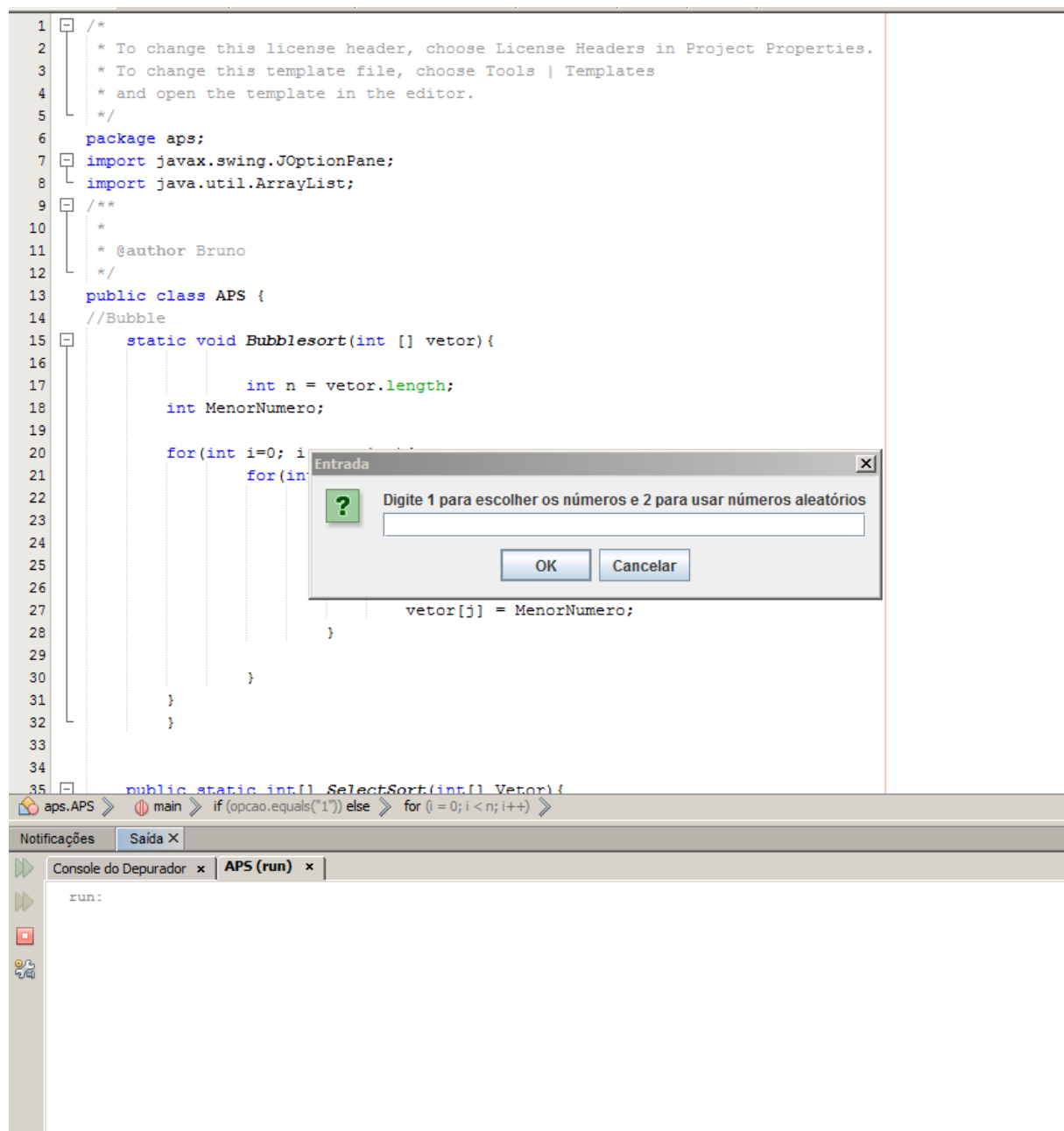
Observações extras

É recomendado que o algoritmo de partição seja usado como uma função separada, por exemplo, a utilização de duas funções, uma de partição e outra de ordenação que utiliza a de partição.

Desenvolvimento

1ºCASO – Obtenção de Dados através do usuário

Na tela inicial do programa, é requisitado o modo de obtenção de dados. Caso digite 1, os dados serão obtidos através da inserção pelo usuário, caso contrário, os dados serão gerados de forma aleatória. No exemplo a seguir, a opção escolhida é a 1.



The screenshot displays an IDE with a Java source file and a runtime dialog box. The code implements a bubble sort algorithm. A dialog box titled 'Entrada' prompts the user to enter '1' for manual input or '2' for random numbers. The IDE's interface includes a package explorer, a breadcrumb trail, a console window, and a debugger window.

```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package aps;
7  import javax.swing.JOptionPane;
8  import java.util.ArrayList;
9  /**
10   *
11   * @author Bruno
12   */
13  public class APS {
14      //Bubble
15      static void Bubblesort(int [] vetor){
16          int n = vetor.length;
17          int MenorNumero;
18          for(int i=0; i<n; i++){
19              for(int j=i+1; j<n; j++){
20                  if(vetor[i]>vetor[j]){
21                      MenorNumero = vetor[j];
22                      vetor[i] = vetor[j];
23                      vetor[j] = MenorNumero;
24                  }
25              }
26          }
27      }
28  }
29
30  public static int[] SelectSort(int[] Vetor){
31      //Bubble
32      static void Bubblesort(int [] vetor){
33          int n = vetor.length;
34          int MenorNumero;
35          for(int i=0; i<n; i++){
36              for(int j=i+1; j<n; j++){
37                  if(vetor[i]>vetor[j]){
38                      MenorNumero = vetor[j];
39                      vetor[i] = vetor[j];
40                      vetor[j] = MenorNumero;
41                  }
42              }
43          }
44      }
45  }
```

Entrada

Digite 1 para escolher os números e 2 para usar números aleatórios

OK Cancelar

aps.APS > main > if (opcao.equals("1")) else > for (i = 0; i < n; i++)

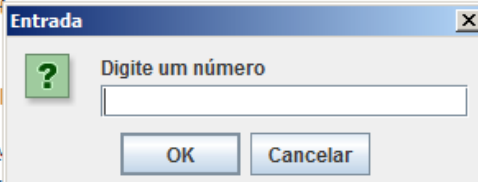
Notificações Saída X

Console do Depurador x APS (run) x

run:

Agora, é requisitado pelo programa que o usuário digite um número, para ser armazenado e depois usado para a ordenação.

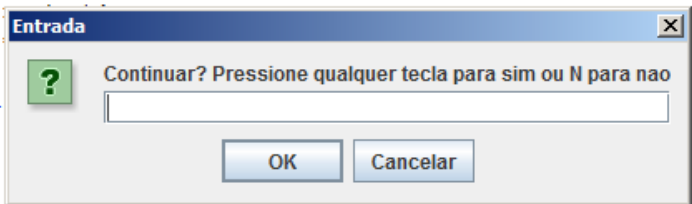
```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package aps;
7  import javax.swing.JOptionPane;
8  import java.util.ArrayList;
9  /**
10 *
11 * @author Bruno
12 */
13 public class APS {
14 //Bubble
15 static void Bubblesort(int [] vetor){
16
17     int n = vetor.length;
18     int MenorNumero;
19
20     for(int i=0; i < n; i++){
21         for(int j=1; j <
22
23             if(vetor
24
25
26
27         vetor[j] = MenorNumero;
28     }
29
30     }
31 }
32 }
33
34
35 public static int[] SelectSort(int[] Vetor){
```



The screenshot shows a Java IDE with a code editor on the left and a console on the right. The code editor displays a Java class named APS with a Bubblesort method. A JOptionPane dialog box titled 'Entrada' is overlaid on the code, prompting the user to 'Digite um número' (Enter a number). The dialog has a green question mark icon and two buttons: 'OK' and 'Cancelar'. The console on the right shows the output 'run:'.

Neste momento, o programa deseja saber se o usuário quer continuar digitando mais valores ou encerrar, caso digite qualquer tecla, o programa irá pedir que digite outro valor, mas se o N for digitado, será encerrado o processo de obtenção de dados.

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package aps;
7  import javax.swing.JOptionPane;
8  import java.util.ArrayList;
9  /**
10   *
11   * @author Bruno
12   */
13  public class APS {
14      //Bubble
15      static void Bubblesort(int [] vetor){
16          int n = vetor.length;
17          int MenorNumero;
18          for(int i=0; i < n; i++){
19              for(int j=i+1; j < n; j++){
20                  if(vetor[i] > vetor[j]){
21                      MenorNumero = vetor[j];
22                      vetor[i] = vetor[j];
23                      vetor[j] = MenorNumero;
24                  }
25              }
26          }
27      }
28  }
29
30  public static int[] SelectSort(int[] Vetor){
31      //Bubble
32      static void Bubblesort(int [] vetor){
33          int n = vetor.length;
34          int MenorNumero;
35          for(int i=0; i < n; i++){
36              for(int j=i+1; j < n; j++){
37                  if(vetor[i] > vetor[j]){
38                      MenorNumero = vetor[j];
39                      vetor[i] = vetor[j];
40                      vetor[j] = MenorNumero;
41                  }
42              }
43          }
44      }
45  }
```



Entrada

Continuar? Pressione qualquer tecla para sim ou N para nao

OK Cancelar

aps.APS > Bubblesort > n >

Notificações Saída X

Console do Depurador x APS (run) x

run:

Caso a opção digitada for N, o usuário será apresentado a essa tela, na qual o programa estará requisitando a opção para selecionar o modo de ordenação em que os valores serão ordenados. Se o usuário digitar 1, será utilizado o Bubble Sort, caso digite 2, será utilizado o Selection Sort e se for digitado o 3, será utilizado o Quick Sort. Nesse caso, a opção informada pelo usuário é o Bubble Sort, de número 1.

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package aps;
7  import javax.swing.JOptionPane;
8  import java.util.ArrayList;
9  /**
10   *
11   * @author Bruno
12   */
13  public class APS {
14      //Bubble
15      static void Bubblesort(int [] vetor){
16          int n = vetor.length;
17          int MenorNumero;
18          for(int i=0; i<n; i++){
19              for(int j=i+1; j<n; j++){
20                  if(vetor[i]>vetor[j]){
21                      MenorNumero = vetor[j];
22                      vetor[i] = vetor[j];
23                      vetor[j] = MenorNumero;
24                  }
25              }
26          }
27      }
28  }
29
30  public static int[] SelectSort(int[] Vetor){
31      //Bubble
32      static void Bubblesort(int [] vetor){
33          int n = vetor.length;
34          int MenorNumero;
35          for(int i=0; i<n; i++){
36              for(int j=i+1; j<n; j++){
37                  if(vetor[i]>vetor[j]){
38                      MenorNumero = vetor[j];
39                      vetor[i] = vetor[j];
40                      vetor[j] = MenorNumero;
41                  }
42              }
43          }
44      }
45  }
```

Entrada

Digite 1 para Bubble Sort, 2 para Selection Sort ou 3 para Quick Sort

OK Cancelar

aps.APS > Bubblesort > n

Notificações Saída X

Console do Depurador x APS (run) x

run:

Após isso, serão apresentados nessa tela final o número total de índices informados pelo usuário, a ordem original de seus valores, a ordem final, obtida após a ordenação pelo método selecionado e o tempo total decorrido durante a ordenação dos dados.

```

1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package aps;
7   import javax.swing.JOptionPane;
8   import java.util.ArrayList;
9   /**
10    *
11    * @author Bruno
12    */
13   public class APS {
14       //Bubble
15       static void Bubblesort(int [] vetor){
16
17           int n = vetor.length;
18           int MenorNumero;
19
20           for(int i=0; i < n; i++){
21               for(int j=1; j < (n-i); j++){
22
23                   if(vetor[i-1] > vetor[j]){

```

aps.APS > Bubblesort > n

Notificações Saída X

Console do Depurador x APS (run) x

```

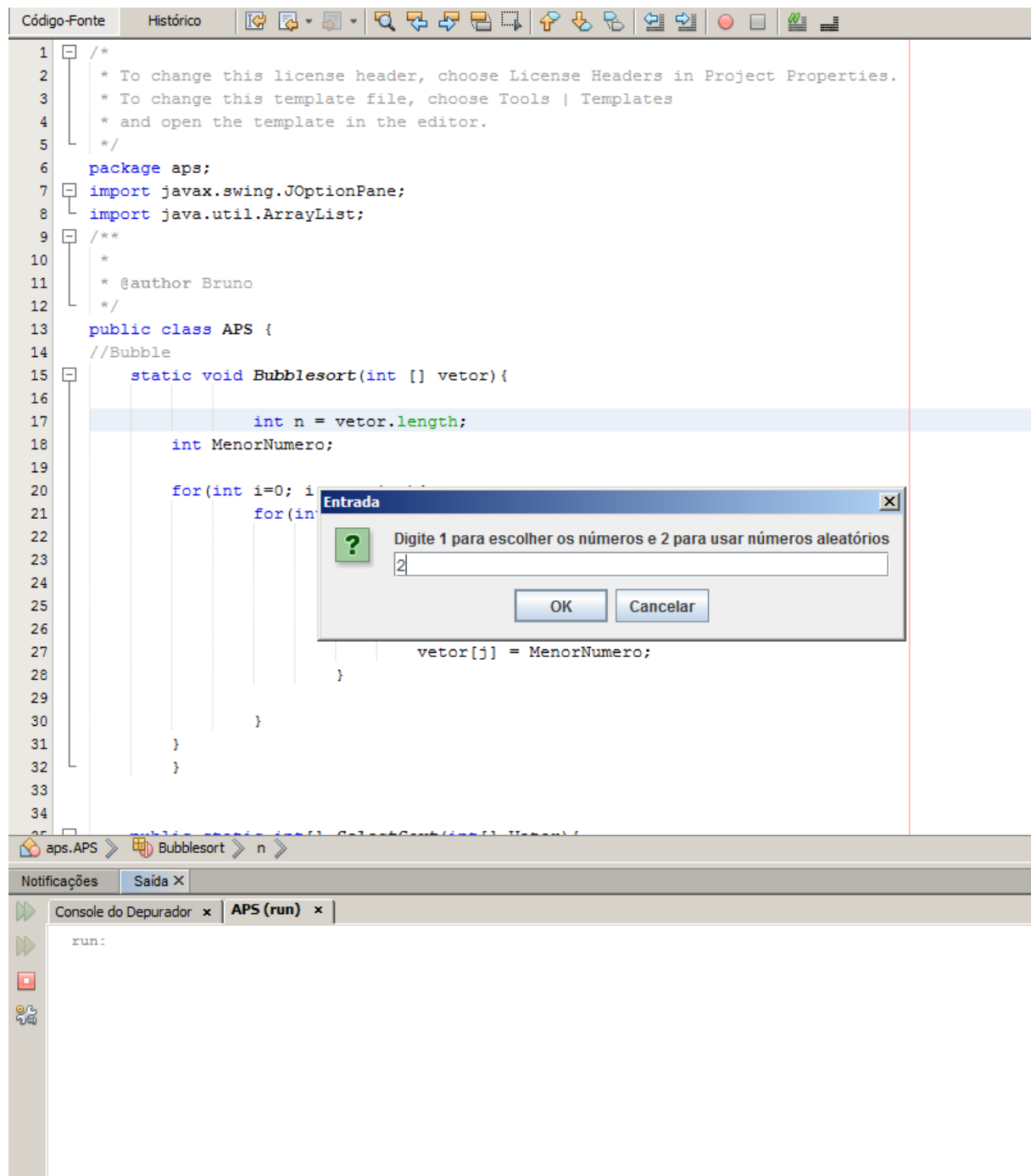
run:
v[i] = Original, Ordenado
-----
v[0] =      5,      4
v[1] =      7,      4
v[2] =      4,      4
v[3] =      8,      5
v[4] =      9,      5
v[5] =      5,      6
v[6] =      4,      7
v[7] =      6,      7
v[8] =    4512,      8
v[9] =      4,      9
v[10] =      7,      9
v[11] =      9,    4512

Tempo de ordenação: 0ms
CONSTRUÍDO COM SUCESSO (tempo total: 3 minutos 34 segundos)

```

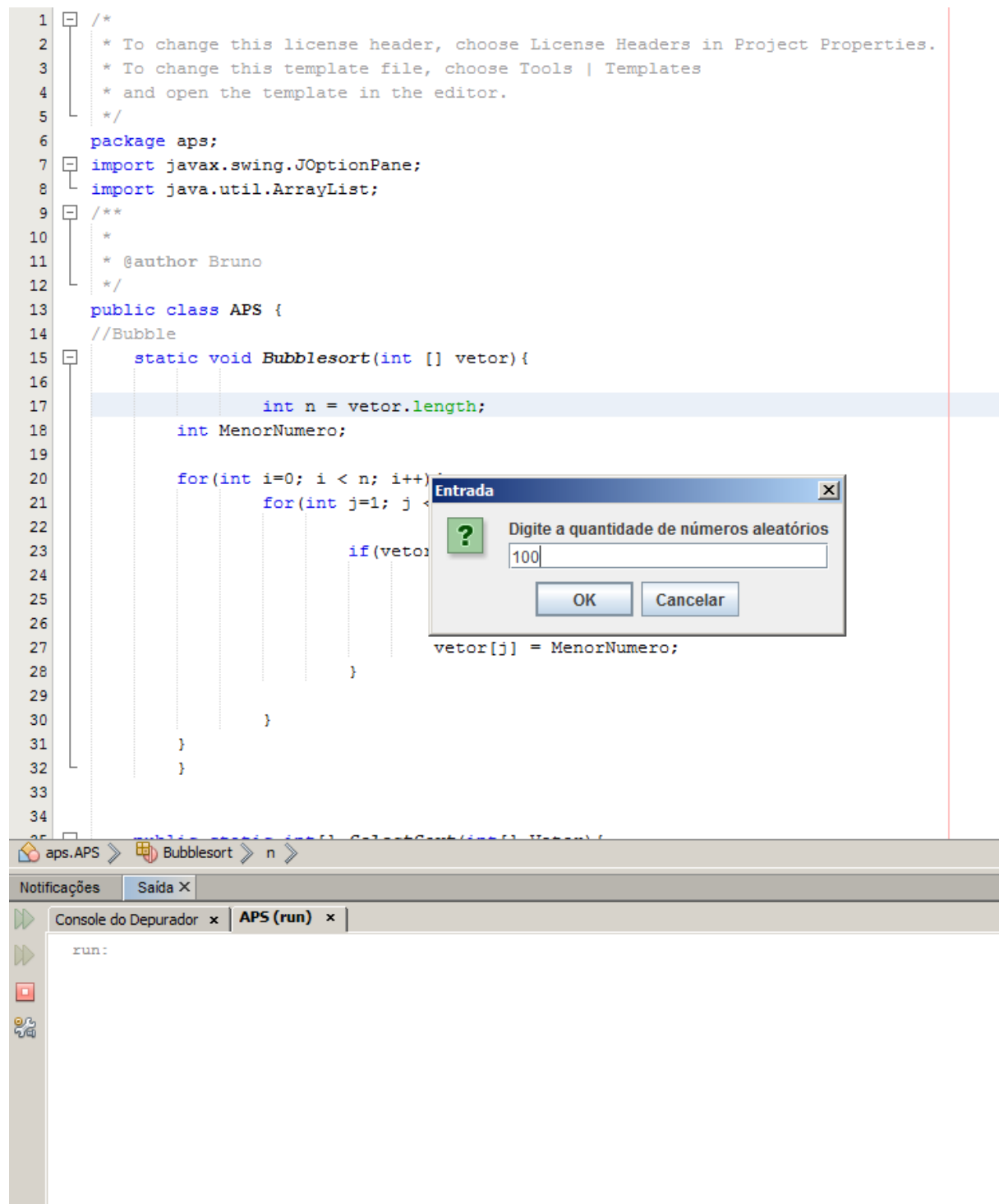
2ºCASO – Dados gerados de forma aleatória pelo programa

Na tela inicial do programa, é requisitado o modo de obtenção de dados. Caso digite 1, os dados serão obtidos através da inserção pelo usuário, caso contrário, os dados serão gerados de forma aleatória. No exemplo a seguir, a opção escolhida é a 2.



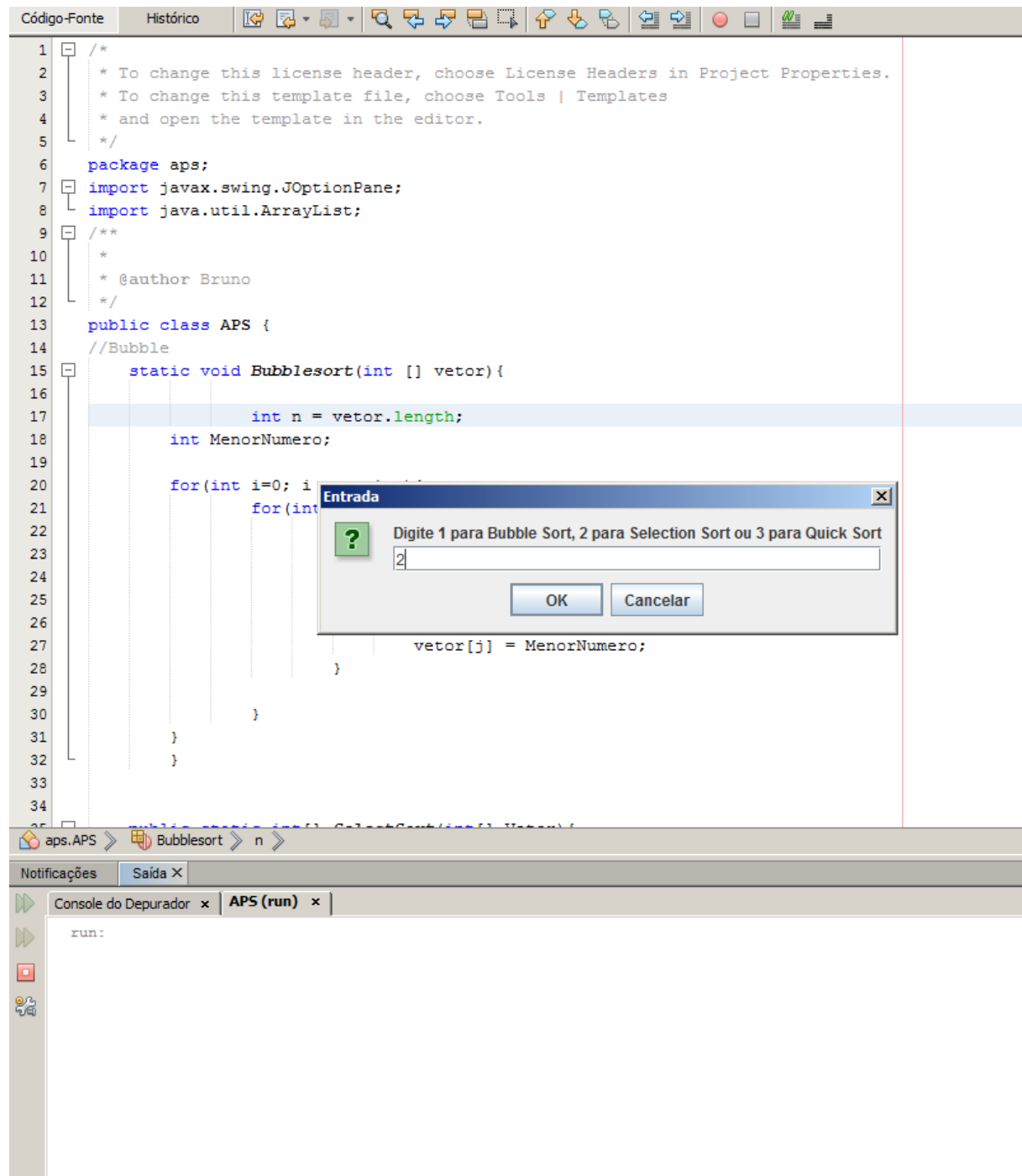
Nesta tela, o programa irá requisitar que o usuário digite a quantidade de números a serem gerados de forma aleatória. Neste exemplo, a quantidade de números a serem gerados será de 100.

```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package aps;
7  import javax.swing.JOptionPane;
8  import java.util.ArrayList;
9  /**
10   *
11   * @author Bruno
12   */
13  public class APS {
14      //Bubble
15      static void Bubblesort(int [] vetor){
16          int n = vetor.length;
17          int MenorNumero;
18          for(int i=0; i < n; i++){
19              for(int j=1; j < n-i; j++){
20                  if(vetor[j] < vetor[j-1]){
21                      int temp = vetor[j];
22                      vetor[j] = vetor[j-1];
23                      vetor[j-1] = temp;
24                  }
25              }
26          }
27      }
28  }
29
30  public static void main(String[] args) {
31      APS obj = new APS();
32      obj.Bubblesort(new ArrayList<>());
33  }
34  }
```



The screenshot shows a Java IDE with a code editor and a console. The code implements a Bubblesort algorithm. A dialog box titled "Entrada" is open, prompting the user to "Digite a quantidade de números aleatórios" (Enter the number of random numbers). The user has entered "100". The dialog has "OK" and "Cancelar" buttons. The IDE interface includes a toolbar at the top, a "Notificações" (Notifications) panel, a "Saída" (Output) panel, and a "Console do Depurador" (Debugger Console) panel. The "Console do Depurador" panel shows the output "run:".

Como no primeiro caso, o programa está requisitando que o usuário informe o método de ordenação que irá ser utilizado para a ordenação do vetor. Se o usuário digitar 1, será utilizado o Bubble Sort, caso digite 2, será utilizado o Selection Sort e se for digitado o 3, será utilizado o Quick Sort. Nesse caso, a opção informada pelo usuário é o Selection Sort, de número 2.



```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package aps;
7  import javax.swing.JOptionPane;
8  import java.util.ArrayList;
9  /**
10   *
11   * @author Bruno
12   */
13  public class APS {
14      //Bubble
15      static void Bubblesort(int [] vetor){
16          int n = vetor.length;
17          int MenorNumero;
18          for(int i=0; i<n; i++){
19              for(int j=i+1; j<n; j++){
20                  if(vetor[i]>vetor[j]){
21                      MenorNumero = vetor[j];
22                      vetor[i] = vetor[j];
23                      vetor[j] = MenorNumero;
24                  }
25              }
26          }
27      }
28  }
29
30  public static int[] SelectSort(int[] Vetor){
31      //Bubble
32      static void Bubblesort(int [] vetor){
33          int n = vetor.length;
34          int MenorNumero;
35          for(int i=0; i<n; i++){
36              for(int j=i+1; j<n; j++){
37                  if(vetor[i]>vetor[j]){
38                      MenorNumero = vetor[j];
39                      vetor[i] = vetor[j];
40                      vetor[j] = MenorNumero;
41                  }
42              }
43          }
44      }
45  }
46  }
```

Entrada

2

OK Cancelar

aps.APS > Bubblesort > n

Notificações Saída X

Console do Depurador x APS (run) x

run:

Por último, como no primeiro caso, nas duas imagens seguintes serão apresentados o número total de índices gerados de forma aleatória, a ordem original de seus valores, a ordem final, obtida após a ordenação pelo método selecionado e o tempo total decorrido durante a ordenação dos dados.

```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */

```

aps.APS > Bubblesort > n >

Notificações Saída X

Console do Depurador x APS (run) x

```

run:
v[i] = Original, Ordenado
-----
v[0] =      52,      3
v[1] =      13,      4
v[2] =      90,      4
v[3] =      70,      5
v[4] =      47,      6
v[5] =      67,      6
v[6] =      93,      7
v[7] =      86,      7
v[8] =      90,      9
v[9] =      28,     13
v[10] =     36,     15
v[11] =     57,     17
v[12] =     51,     18
v[13] =     70,     18
v[14] =     65,     19
v[15] =     62,     19
v[16] =     53,     21
v[17] =    100,     22
v[18] =      3,     22
v[19] =     29,     24
v[20] =     15,     25
v[21] =     72,     26
v[22] =     81,     26
v[23] =     26,     27
v[24] =     79,     28
v[25] =      4,     28
v[26] =     37,     29
v[27] =     18,     31
v[28] =     26,     32
v[29] =     82,     33
v[30] =     83,     36
v[31] =     71,     37
v[32] =     19,     37
v[33] =     50,     38
v[34] =      7,     43
v[35] =     88,     44
v[36] =     25,     46
v[37] =     38,     47
v[38] =     75,     47
v[39] =     81,     48
v[40] =     22,     49
v[41] =     74,     50
v[42] =     32,     51

```

```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */

```

aps.APS > Bubblesort > n >

Notificações Saída X

Console do Depurador x APS (run) x

v[57] =	54,	67
v[58] =	67,	69
v[59] =	22,	69
v[60] =	78,	70
v[61] =	18,	70
v[62] =	33,	71
v[63] =	99,	72
v[64] =	73,	73
v[65] =	62,	73
v[66] =	67,	74
v[67] =	24,	75
v[68] =	69,	78
v[69] =	48,	78
v[70] =	90,	79
v[71] =	73,	81
v[72] =	52,	81
v[73] =	7,	81
v[74] =	17,	81
v[75] =	69,	81
v[76] =	46,	82
v[77] =	9,	83
v[78] =	100,	84
v[79] =	47,	84
v[80] =	92,	85
v[81] =	56,	86
v[82] =	84,	86
v[83] =	94,	88
v[84] =	6,	88
v[85] =	81,	88
v[86] =	37,	90
v[87] =	27,	90
v[88] =	88,	90
v[89] =	5,	90
v[90] =	95,	91
v[91] =	81,	92
v[92] =	4,	92
v[93] =	54,	93
v[94] =	78,	94
v[95] =	49,	95
v[96] =	6,	99
v[97] =	58,	99
v[98] =	88,	100
v[99] =	86,	100

Tempo de ordenação: 0ms
CONSTRUÍDO COM SUCESSO (tempo total: 1 minuto 41 segundos)

A apresentação dos resultados comparativos entre os métodos e diferentes casos estará contida no próximo tópico.

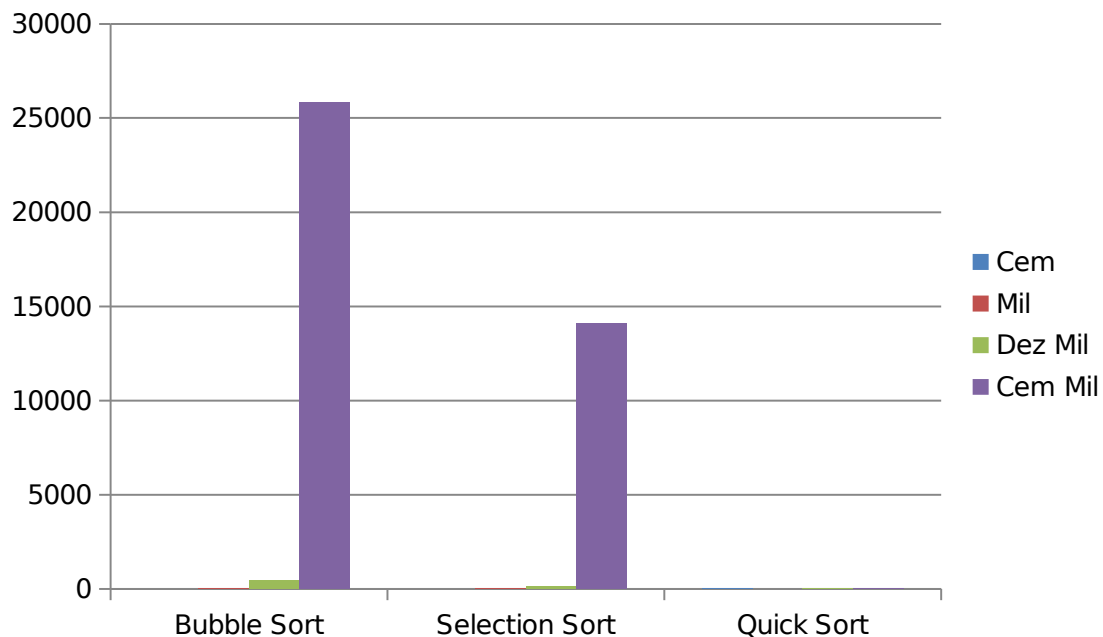
Resultados e Discussão

Os resultados que serão apresentados a seguir foram obtidos através de um computador com processador Dual-Core 2.5GHz e memória RAM DDR2 de 4GB.

Dados Aleatórios

Aleatórios	Cem	Mil	Dez Mil	Cem Mil
Bubble Sort	0	39	462	25835
Selection Sort	0	5	155	14087
Quick Sort	1	0	4	35

(Valores na escala de milisegundos (ms))



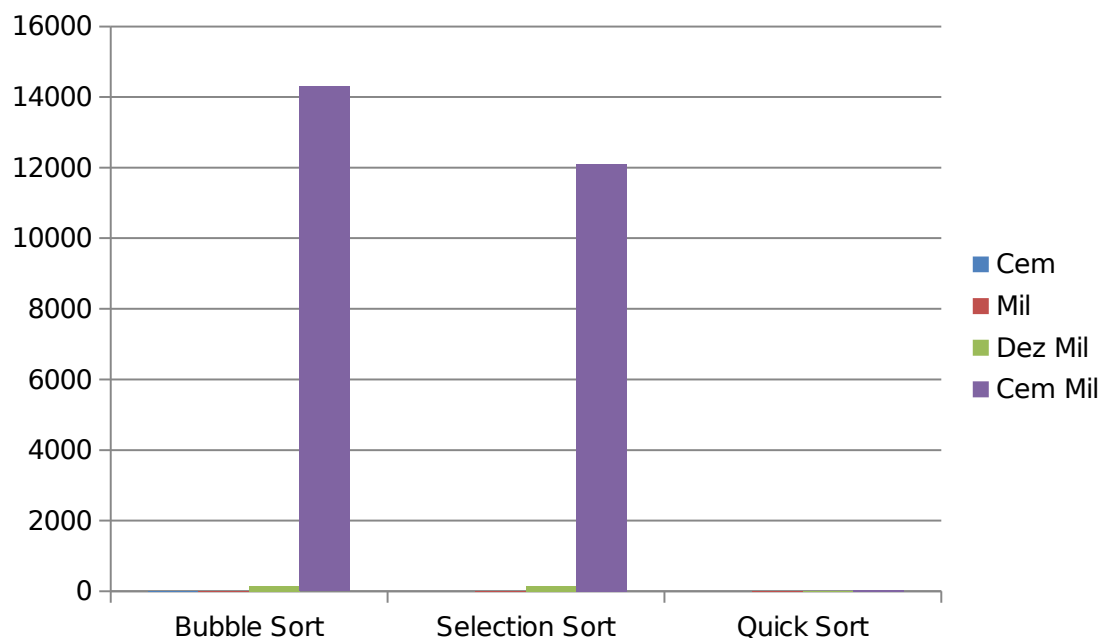
Na condição de dados em ordem aleatória, o Bubble Sort obteve de longe o pior desempenho, 8 vezes mais demorado para ordenar mil números, 3 vezes para ordenar dez mil e quase duas vezes mais para ordenar cem mil em comparação

com o segundo da lista, o Selection Sort. E como melhor desempenho bem acima dos demais se encontra o Quick Sort.

Dados pré-ordenados em ordem decrescente

Decrescente	Cem	Mil	Dez Mil	Cem Mil
Bubble Sort	1	12	154	14303
Selection Sort	0	9	143	12096
Quick Sort	0	1	11	33

(Valores na escala de milisegundos (ms))

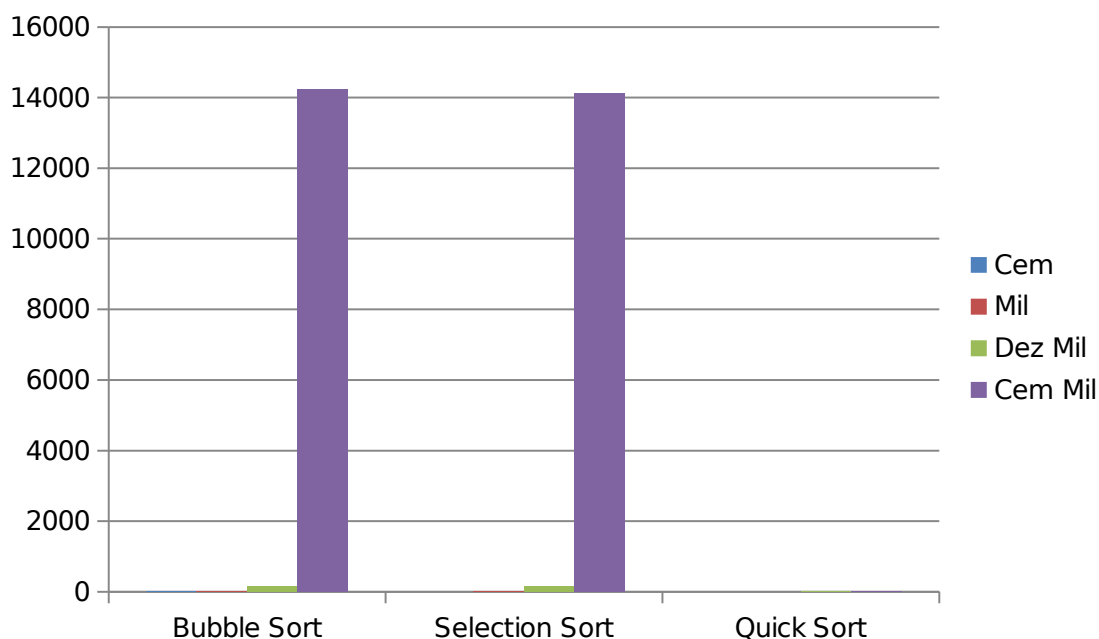


Na condição de dados pré-ordenados em ordem decrescente, o Bubble Sort obteve o pior resultado, mas desta vez não tão ruim como a primeira, ficando apenas um pouco atrás do segundo da lista, o Selection Sort. E como melhor desempenho bem acima dos demais novamente, se encontra o Quick Sort.

Dados já ordenados (ordem crescente)

Já ordenado	Cem	Mil	Dez Mil	Cem Mil
Bubble Sort	1	7	163	14226
Selection Sort	0	7	156	14126
Quick Sort	0	0	4	14

(Valores na escala de milissegundos (ms))



Na condição de dados já ordenados, o Bubble Sort obteve o pior resultado, mas como no caso anterior, ficando apenas um pouco atrás do segundo da lista, na verdade, quase levou o mesmo tempo que o Selection Sort. E como melhor desempenho bem acima dos demais novamente, se encontra o Quick Sort.

Em relação aos testes:

O Quick Sort é o mais rápido para todos os tamanhos e tipos de dados experimentados em relação aos outros. Em dados do mesmo tamanho, o Quick Sort executa mais rápido para dados ordenados, o Selection só ganha ou empata com o Quick Sort em todos os casos em que o tamanho de dados foi de 100. O Bubble

Sort se demonstra o pior nos casos em ordem aleatória e decrescente, sendo semelhante ao Selection apenas nos dados já ordenados.

Considerações Finais

No trabalho foi apresentado o que são métodos de ordenação, alguns dos mais conhecidos, foi explicado como funcionam, seus desempenhos e foram também aplicados na prática em um programa. Em relação aos desempenhos dos métodos, o quick sort é o algoritmo mais eficiente que existe para uma grande variedade de situações, é recursivo, o que demanda uma pequena quantidade de memória adicional.

Referências Bibliográficas

<http://www.algolist.net/Algorithms/>

<http://wiki.icmc.usp.br/images/b/b3/SCC501Cap4.pdf>

<http://homepages.dcc.ufmg.br/~cunha/teaching/20121/aeds2/sortingcmp.pdf>

<http://opendevnotes.blogspot.com.br/2008/09/java-medir-tempo-de-execuo-de-tarefas.html>

Código Fonte

Classe principal: APS

```
/*  
 * To change this license header, choose License Headers in Project Properties.  
 * To change this template file, choose Tools | Templates  
 * and open the template in the editor.  
 */  
  
package aps;  
  
import javax.swing.JOptionPane;  
  
import java.util.ArrayList;  
  
/**  
 *  
 * @author Bruno  
 */  
  
public class APS {  
  
    //Bubble  
  
    static void Bubblesort(int [] vetor){
```

```

        int n = vetor.length;

        int MenorNumero;

        for(int i=0; i < n; i++){

            for(int j=1; j < (n-i); j++){

                if(vetor[j-1] > vetor[j]){

                    MenorNumero = vetor[j-1];

                    vetor[j-1] = vetor[j];

                    vetor[j] = MenorNumero;

                }

            }

        }

    }
}

```

```

public static int[] SelectSort(int[] Vetor){

    for (int i = 0; i < Vetor.length - 1; i++)

    {

        int aux = i;

        for (int j = i + 1; j < Vetor.length; j++)

            if (Vetor[j] < Vetor[aux])

```

```

        aux = j;

        int MenorNumero = Vetor[aux];

        Vetor[aux] = Vetor[i];

        Vetor[i] = MenorNumero;
    }

    return Vetor;

}

```

//QuickSort e seus 2 métodos

```

int dividir(int vetor[], int esq, int dir)
{
    int i = esq, j = dir;

    int tmp;

    int pivot = vetor[(esq + dir) / 2];

    while (i <= j) {
        while (vetor[i] < pivot)
            i++;

        while (vetor[j] > pivot)
            j--;
    }
}

```



```

        if (i <= j) {
            tmp = vetor[i];
            vetor[i] = vetor[j];
            vetor[j] = tmp;

            i++;
            j--;
        }

    }

    return i;
}

void quickSort(int vetor[], int esq, int dir) {
    int index = dividir(vetor, esq, dir);

    if (esq < index - 1) {
        quickSort(vetor, esq, index - 1);
    }

    if (index < dir) {
        quickSort(vetor, index, dir);
    }
}

```

```

public static void main(String[] args) {

    String continuar="", opcao="";

    int contador =0, i ,n;

    long time = 0;

    ArrayList vetororiginal = new ArrayList();


    do {

        opcao =JOptionPane.showInputDialog("Digite 1 para escolher os números e 2
para usar números aleatórios");

        if (!opcao.equals("1") && !opcao.equals("2")) {

            JOptionPane.showMessageDialog(null, "Opção inexistente, digite
novamente");

        }

    } while (!opcao.equals("1") && !opcao.equals("2"));


    if (opcao.equals("1")) {

```

```

while (!continuar.equals("n")) {

    vetororiginal.add(Integer.parseInt(JOptionPane.showInputDialog("Digite um
número"))));

    contador++;

    continuar = JOptionPane.showInputDialog("Continuar? Pressione qualquer
tecla para sim ou N para nao");

}

n = contador;

int vetorordenado[] = new int[n];

for (i = 0; i < n; i++) {

    vetorordenado[i] = Integer.parseInt(vetororiginal.get(i).toString());

}

do {

    opcao = JOptionPane.showInputDialog("Digite 1 para Bubble Sort, 2 para
Selection Sort ou 3 para Quick Sort");

    if (!opcao.equals("1") && !opcao.equals("2") && !opcao.equals("3")) {

        JOptionPane.showMessageDialog(null, "Opção inexistente, digite
novamente");

    }

} while (!opcao.equals("1") && !opcao.equals("2") && !opcao.equals("3"));

```

```

if (opcao.equals("1")) {

    Chronometer.start();

    Bubblesort(vetorordenado);

    Chronometer.stop();

    time = Chronometer.elapsedTime();

}

else {

    if (opcao.equals("2")) {

        Chronometer.start();

        SelectSort(vetorordenado);

        Chronometer.stop();

        time = Chronometer.elapsedTime();

    }

    else {

        APS ordenarQuick = new APS();

        Chronometer.start();

        ordenarQuick.quickSort(vetorordenado, 0, n-1);

        Chronometer.stop();

        time = Chronometer.elapsedTime();

    }

```

```

    }

    System.out.println("v[i] = Original, Ordenado");

    System.out.println("-----");

    for (i=0; i<vetororiginal.size(); i++) {

        System.out.printf("v[%d] = %8s, %8d\n",i, vetororiginal.get(i).toString(),
vetorordenado[i]);

    }

    System.out.println("");

    System.out.println("Tempo de ordenação: " + time +"ms");

}

//aleatorio

else {

    n = Integer.parseInt(JOptionPane.showInputDialog("Digite a quantidade de
números aleatórios"));

    int x;

    int vetorordenado[];

    vetorordenado = new int[n];

    for (i=0; i<n; i++) {

```

```
x = (int)Math.round(Math.random()*n); //o resultado desta expressao sera um
numero
```

```
vetororiginal.add(x) ; // no intervalo de 0 ate 100
```

```
vetorordenado[i] = x;
```

```
}
```

```
do {
```

```
    opcao = JOptionPane.showInputDialog("Digite 1 para Bubble Sort, 2 para
Selection Sort ou 3 para Quick Sort");
```

```
    if (!opcao.equals("1") && !opcao.equals("2") && !opcao.equals("3")) {
```

```
        JOptionPane.showMessageDialog(null, "Opção inexistente, digite
novamente");
```

```
    }
```

```
    } while (!opcao.equals("1") && !opcao.equals("2") && !opcao.equals("3"));
```

```
if (opcao.equals("1")) {
```

```
    Chronometer.start();
```

```
    Bubblesort(vetorordenado);
```

```
    Chronometer.stop();
```

```
    time = Chronometer.elapsedTime();
```

```

    }

    else {

        if (opcao.equals("2")) {

            Chronometer.start();

            SelectSort(vetorordenado);

            Chronometer.stop();

            time = Chronometer.elapsedTime();

        }

        else {

            APS ordenarQuick = new APS();

            Chronometer.start();

            ordenarQuick.quickSort(vetorordenado, 0, n-1);

            Chronometer.stop();

            time = Chronometer.elapsedTime();

        }

    }

    System.out.println("v[i] = Original, Ordenado");

    System.out.println("-----");

    for (i=0; i<vetororiginal.size(); i++) {

        System.out.printf("v[%d] = %8s, %8d\n",i, vetororiginal.get(i).toString(),
vetorordenado[i]);

```

```
}
```

```
System.out.println("");
```

```
System.out.println("Tempo de ordenação: " + time + "ms");
```

```
}
```

```
}
```

```
}
```

Classe utilizada para cronometrar o tempo de ordenação:

```
/*
```

* To change this license header, choose License Headers in [Página 40 de 42](#)
Project Properties.

* To change this template file, choose Tools | Templates

* and open the template in the editor.

*/

package aps;

/**

*

* @author Bruno

*/

public final class Chronometer {

private static long startValue;

private static long stopValue;

private static long timeDiff;

/**

* Inicia a contagem temporal

*/

public static void start() {

startValue = System.currentTimeMillis();

stopValue = 0;

timeDiff = 0;

}

/**

* Calcula a diferença temporal

```

*/

public static void stop() {

    stopValue = System.currentTimeMillis();

    timeDiff = stopValue - startValue;

}

/**

* Retorna o diferença de tempo medida

* @return tempo em milisegundos

*/

public static long elapsedTime() {

    return timeDiff;

}

}

```